RECEIVED
JAN 13 1981
PETER BONFIELD

M E M O R A N D U M
January 8, 1981

TI STRICTLY PRIVATE

COPY HERZOG
C.B.
INFO
+RETN

TO: Johnny Barrett

CC: Pete Bonfield ✓
Dan Enzone
Jerry Junkins
Mike Lockerd
Jim Oliphant
Bill Sick
Darrell Whitten

FROM: Tony Barlow

SUBJECT: TI-88 CRITIQUE: READER'S DIGEST VERSION

Recognizing the tight schedule you are working to and the fact that you may wish to make some changes in the TI-88 as a result of my comments, I am providing in this memo an unpolished report now in preference to a more literary one later. You may still have the Pulitzer Prize version later if need be. Before taking up specifics, allow me some religious observations.

As we discussed, I am disappointed in the TI-88 and believe that it is fundamentally the wrong product for introduction into the market at the wrong time. The problem is that we are past the HP-65 days when memory size was barely adequate and only then so with efficient use of the instruction word. Now there is no great premium on memory capacity. The TI-59 memory is already so large that filling it tends to produce a program/data structure which requires many minutes to execute. Increased memory capacity, therefore, only is of benefit if it is attended by commensurate increases in the other resources (throughput and I/O rate) or if it is used for another purpose entirely, namely to reduce complexity of operation. As the memory increases, less efficient programming techniques are feasible. We have already seen the introduction of the BASIC handheld programmables as an example of this. I imagine you are weary of hearing about the Sharp 1210/1211 calculators, so let me assure you that I am not saying that BASIC is the answer to the world's requirements for handheld programmables. What I am saying is that the Sharp 1210/1211 is responsive to a need, to the need most typical at the present stage of handheld programmables: reducing the complexity of programming. I perceive a fundamental difference between the purpose of a handheld programmable calculator and a hand-sized computer which as far as I'm concerned provides the <u>raison d'ete</u> for an HP-65, SR-52 or Sharp 1211. The handheld programmables enable the user to get the answer to a problem which requires on or two inputs, the answer expressible in terms of one or two outputs, quickly, while the problem is still fresh, during a meeting for example.

The facts that (1) the programs used to perform this magic may have been written ahead of time in anticipation of the need and (2) the programmable calculator may be used to emulate a computer, processing many inputs to output (i.e., print) many output numbers, while valid extensions of the utility of the handheld programmable, also obscure its unique function: to enable the user to program on the spot so as to achieve immediate answers to short problems. Of course, immediate answers to problems are only of value if they are correct; hence the high premium I place on simple, reliable operation. Programming the TI-88 should be so simple that after a short acquaintance with the machine, the user feels comfortable that he can get the right answer the very first time. In contrast to this objective, I got my first correct answer to a 60-step program only after four hours of time on the TI-88, following almost one week of study of the owner's manual draft and considerable experience with the HP-65, HP-67, SR-52, TI-59 and Sharp 1211. In fact, the TI-88 is the most whimsical, arbitrary, difficult programmable device I have ever attempted to work with. As I told you before, one thought which comforts me is that I won't ever have to use it after this job is complete. Such a statement does not seem to admit any qualifications, but in fact there are several: first, I only refer to the functionality of the TI-88 from the programming point of view. Once the program has been successfully written (purchased as Solid-State Software, for example), the prompting capability allowed by the alphanumerics should be a boon to the end user and the product extremely easy to use. Second, the experience I claimed with various handhelds, the uses I put them to in my work are not completely beneficial toward reaching a positive opinion of the TI-88. To some extent, my experience with the TI-59 was counterproductive, causing me to underestimate the challenge of the TI-88. (However, this is a problem which will be shared by a sizeable portion of your potential market.) The third qualification is that I did not have the benefit of finished owner's manual. For this product especially you will require an excellent owner's manual which serves two purposes, initial training and quick reference. The final qualification to my worst-ever remark is of most short-term significance. Now that I have actually successfully run one program, I am "feeling my oats" and have several suggestions for changing the TI-88 so as to improve its functionality. While the result would not win any prizes from me, I do believe these changes would make the product viable. These suggestions comprise the second portion of this memo. Before taking up these suggestions, allow me to close this section with a couple of more philosophical remarks about the TI-88 overall.

TI STRICTLY PRIVATE

Although I have dumped on the TI-88 pretty hard, there were several features which I liked very much which I would like to name here. I have already mentioned the alphanumeric capability; one has only to use this feature (in the program-written-in-anticipation-of-need environment) to appreciate the self-documenting nature of this feature. Actually, this feature is of less importance to me than three others. The best thing about the TI-88 vs. the TI-59 is its long battery life and LCD display. (It is hard to feel comfortable carrying a TI-59 to a meeting if the choice is between an extension cord and a battery which dies in midflight.) The next most attractive feature is the non-volatile memory and the third is the ability via the RAM modules to customize one's own "Solid-State Software." In effect, the public can be told they do not need to wait for TI SSS modules to address their needs; they can make their own out of their favorite programs provided by themselves or (eventually) user exchange groups. While cassette loading of such modules is attractive, the cassette does not, in my opinion, replace the function of a card reader and the omission of card-read capability on the TI-88 represents to me a serious deficiency shared with the Sharp 1211.

Finally, I thank you for lending me a copy of your market survey. I found it interesting and believe such surveys can provide a useful input for guiding product design. On the other hand, I think the results of such surveys are subject to more than one interpretation and should always be used in concert with other judgments and procedures. I understand that you are in complete agreement with this point of view. Perhaps some time in the future after the severe time pressures of product introduction are behind you (are they ever in Consumer Products?) we could chat about possible ways of improving the quality of these surveys. Now let me turn to my shopping list of suggestions for improving the TI-88 in the next six weeks or so.

The first three items on my list are absoltely essential -- there is no option. They are:

1. Eliminate implied multiplication.
2. Eliminate implied multiplication.
3. Eliminate implied multiplication.

he did this to the SR-52 originally — his fetish or we <u>won't learn</u>? CB

Implied multiplication should be killed, eliminated, expunged, purged, cremated, buried and forgotten. Even in a language such as BASIC, in which transfers are constrained, implied multiplication complicates the heirarchical rules and causes mistakes; in the more flexible, quasi-assembly language of the TI-59 and TI-88, it introduces exquisite complexity. Even your own algorithm designers are not sure how it works. Merely providing the option of turning implied multiplication on or off is disaster in the making. If you must retain it in the Equation mode where it is relatively benign, then fine; but for heaven's sake get rid of it in the programming mode.

I also strongly recommend that you eliminate the unary minus. How can we go to the public with a product which requires three different ways of executing "minus" (unary minus, the subtract operator, and the post fix change-sign) when our competitors can get by with one? If the display requirements of the Equation mode are perceived to include a leading minus, then it should be obtained using the subtract operator. Surely your algorithm can be changed to provide for an error-free leading minus in Equation mode without introducing the unary operator explicitly.

Next, I would simplify operation in the Equation mode by requiring that all variable definition statements precede evaluation of expressions. In other words, the structure should be:

```
(variable definition statement)
               o
               o
               o
(variable definition statement)
(expression evaluation).
```

The variable definition should function as currently designed; namely, the present value should be displayed and the option to either enter a new value or continue should be provided. The EVAL key should be renamed (START or BEGIN) to avoid the present logical inconsistency that in operation, data entry begins in response to pressing EVAL and evaluation takes place after data entry.

The prefixed unary functions introduce further complexity into the heirarchy and are an unnecessary and ineffective sop to the banner of making the program resemble the algebraic text. Even with prefixed unary functions, the TI-88 code does not look much like text (what with RCL022, EXC101, ST: 16 and the like). Prefixed unaries just make mistakes more likely. What is important to the user is whether the program can easily be written in the first place and whether it can be read, interpreted and perhaps modified six months later. The TI AOS, Sharp's BASIC and Hewlett-Packard's RPN all satisfy the first requirement (yes Virginia, they all do). In readability after the fact, RPN goes down in flames but AOS is almost as readable

TI STRICTLY PRIVATE

as BASIC. The proposed EOS alterations just turn a very passable linebacker into a very poor pitcher. If you absolutely must retain prefixed unaries for cosmetic reasons in the Equation mode and if this requires their retention in the programming mode, then at least discourage the user from using them without parentheses: SIN (RCL 1) + RCL 2 rather than SIN RCL 1 + RCL 2, for example. Explain in the manual that omission of parentheses will not result in an error code; explain in an Appendix to the advanced user how the heirarchy is modified by the unary functions, but discourage the omission of parentheses except where the result is very easily understood (as in SIN 30 =) to avoid errors of programming. The whiz kids can then omit parentheses (without our complicity) as befits their comprehension of the system without burdening the low-speed luggers with greater heirarchical complexity and reduced reliability.

Attempting to resume execution past a Run/Stop in a plug-in module resident program can lead to an error-stop and puzzeling message ("Cue or R/S mode?") well after the fact. Instead, the error condition and its message should be eliminated with user-beware in effect as far as unsatisfied subroutine returns are concerned. I'm not sure I like your practice of handling module execution as a subroutine call, but I don't know what to do about that at the moment.

The alphanumerics are pretty easy to use. I do object to what occurs in a formatted display (using the block command) when the number in the display register will not fit the allocated space. In particular, when E-format is required for its representation, the displayed number is incorrect and no warning results. For example, anticipating that four places are adequate to display the answer would lead to something like: P = -1.6 UNITS for a value of 1.67325 EE + 18. Either an error code (format exceeded) should be given or else the alphanumeric should be omitted and the number displayed alone when the format overflow occurs.

The result of underflow during execution should result in the quantity being replaced with zero. At least this should be an option.

Currently, the NOP is a pure NOP. It does not even hold its place. For example, the sequence STO 2 NOP NOP NOP 3.5 + ... results in a store to register 23. I believe it is preferable for NOP to terminate address entry for both register and program addresses.

TI STRICTLY PRIVATE

A display indication of keyboard shift status should be provided. The present IVV status indication can be eliminated for this purpose.

Holding down the arrow-right and arrow-left keys should scroll the cursor. Single-step scrolling is not sufficient.

The ENT/CONT branch (OP 06) does not eliminate the benefit of a dedicated flag which sets on data entry (keyboard) and is reset after any run-mode operation.

Assuming you remove unary minus, replace it with flip flag. (The present INV RST is obtuse.)

If a composite test can be made up such as (IF A + IF = B) for "IF (display) is greater than A or if it is = B then" and the analogous (IF A * IF = B) which replaces "OR" with "AND" then multiple tests would be much simpler. The present situation whereby two conditionals back-to-back are interpreted with an OR is much less satisfactory.

Finally, the time/date with all its associated paraphenalia and queries as to "Do you want to set me now?" are cheap gimmicks. I own a watch.

I will close this memo as I began it. Remember that greater complexity is more tolerable in an ASC than in a home computer and greater complexity is more tolerable in a home computer than in a handheld calculator. With that epitaph, let me assure you that I will be glad to discuss this memo or other matters relating to the TI-88 further with you. On the other hand, I recommend that you bring in a few other folk as you did me for independent appraisal if your time permits. (Names which come to my mind are Sid Nolte, Clif Penn, Carlisle Phillips, Al Riccomi be engineers like Jack Smith, Jim Couvillon, all-round smart guys like Mike O'Hagan and Doug Ziemer and maybe some bean counting types as well.)

All negative remarks notwithstanding, I wish you best of luck with the eventual TI-88 and hope that it will make you all rich and famous.

Tony Barlow
TB:pn

(DWW/27)